EXPLODING BIPLOTS WITH DENSITY AXES IN PLOTLY

Delia Sandilands

Assignment presented in partial furfillment
of the requirement for the degree
MCom (Financial Risk Management)
at the University of Stellenbosch

Supervisor: Dr. C.J. van der Merwe

Degree of confidentiality: A

December 2020

ACKNOWLEDGEMENTS

Thank you to all those who gave helpful insight, a listening ear, and an encouraging word throughout this process. Amongst these, I would especially like to extend my gratitude to:

- firstly, my supervisor Dr. van der Merwe, without his insightful feedback, directive suggestions and help, this work would not have been possible.
- Adriaan Rowan and Prof Lubbe that contributed their time and helpful recommendations.
- my family and flatmate, whose support was invaluable during this year and
- finally, to God, for without Him I can do nothing.

SUMMARY

Biplots are visual approximations to multidimensional data sets and are very useful in data driven procedures. However, certain issues occur when dealing with a large number of observations or variables. A proposed solution to the former would be moving the variable axes orthogonally to the edges of the plot with an automatic procedure. This unclutters the center of the plot and makes it easier to interpret specific observations. When dealing with a large number of observations the datapoints become indifferentiable from one another and this could be difficult to interpret. This problem was addressed by creating densities on the variable axes. Along with these proposed enhancements, an interactive element to biplot plotting was introduced via the use of `Plotly` in `R`. This allows the user to dissect the biplot even further.

A web-based Shiny application was built as supplement to this paper, and can be found at https://carelvdmerwe.shinyapps.io/ExplodingBiplots/. Here the user can interact with the data sets, proposed methodology, and functionalities presented in this research.

OPSOMMING

'n Bi-stipping konstruksietegniek is baie voordelig wanneer veelvoudige data gevisualiseer moet word. Hierdie tegniek het egter 'n paar tekortkominge wanneer datastelle baie veranderlikes of observasies het. Wanneer die datastel baie veranderlikes het, raak die bi-stipping oor vol en dit raak al hoe moeiliker om te onderskei tussen veranderlikes se onderskeie asse. 'n Outomatiese algoritme om die asse te skuif word voorgestel om hierdie probleem aan te spreek. In de geval van baie observasies word dit moeilik om die verskillende gevalle of die waarskynlikheidsdigtheid van die observasies van die verskeie veranderlikes te sien. 'n Moontlike oplossing hiervoor wat insluit die konstruksie van waarskynlikheidsdigthede op elke veranderlike se as is voorgestel.

'n Web-gebaseerde Shiny toepassing was gebou aanvullend tot hierdie artikel, en kan gevind word op die webtuiste https://carelvdmerwe.shinyapps.io/ExplodingBiplots/. Hier kan die gebruiker self die toepassing van die prosesse voorgestel in hierdie navorsing sien.

SAQA OUTCOMES

For this masters assignment an article based approach was followed. This entailed producing a working paper (provided in this assignment document) which was given a provisional mark (on submitting to a journal) and corrections by the examiner. These corrections were incorporated and the paper was submitted to a journal that was agreed upon by the student and supervisor. To ensure compliance of the outcomes of the assignment, a list of all the required South African Qualification Authority (SAQA) outcomes[1] are listed below and how they were achieved in this assignment.

Outcome	Fulfilment
Scope of knowledge	Various visual enhancements for biplots were developed.
Knowledge literacy	Literature and available packages were examined to find what packages are available for constructing biplots and how they have been enhanced.
Method and procedure	All underlying theory and processes are presented in detail.
Problem-solving	Two algorithms were developed, one for the translating of biplots axes and one for creating densities on the biplot axes. These methods involved coding in R, and various other geometric mathematics.
Ethics and professional practice	Careful consideration was given in obtaining and using the data. Professionalism was upheld to the highest standard at all times. Furthermore, all required ethical clearances were obtained where necessary.
Accessing, processing and managing information	Various packages for producing biplots in R were examined to see what enhancements have been proposed for solving two main issues, overcrowding of observations and the overlapping of biplot axes. These were then used to develop enhancements for biplots.
Producing and communicating information	The target audience was always taken into account during the writing of the article.
Context and systems	The algorithms produced for this paper was implemented in R to illustrate the effectiveness thereof.
Management of learning	Several consulting sessions with the learner's supervisor aided in promoting self learning strategies. These sessions allowed the learner to gain independence with regards to his learning and enabled them to enhance their skills to work in a professional environment.
Accountability	The paper was written independently by the learner incorporating suggestions under the guidance of their supervisor.

[1] See: https://www.saqa.org.za/docs/misc/2012/level_descriptors.pdf

PLAGIARISM DECLARATION

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. Accordingly, all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
4. I also understand that direct translations are plagiarism.
5. I declare that the work contained in this assignment, except otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this assignment or another assignment.

Student number	Signature
D. Sandilands	19 October 2020
Initials and surname	Date

TABLE OF CONTENTS

EXPLODING BIPLOTS WITH DENSITY AXES IN PLOTLY

A version of this paper was submitted to an international statistical journal.

Abstract

Biplots are useful when visualizing multivariate data. It can, however, sometimes be challenging to interpret, for example when the axes and points cause overcrowding of the plot. This overcrowding is often due to the presence of many variables, highly correlated variables, or merely data sets with a large number of observations. In this paper improvements to the biplot are made to address these shortcomings. These improvements include: i) the automatic parallel translation, or "explosion", of axes, ii) the use of densities on the axes to improve interpretation and representation of large data sets, and iii) introducing interactive biplots via the use of the `Plotly` package in `R`. These improvements result in a better composition of the plot to make it seem less crowded, more easily interpretable, offer additional information that can get lost in the case of a high volume of data, and allowing the user to inspect the biplot element-wise.

Keywords: Automated Axis Translation, Interactivity, Multidimensional Data Visualization

1. Introduction

Methods for visualizing multivariate data often only allow one to see a pairwise relationship between variables. Scatterplots, for example, show correlations and trends between each pair of variables in the data set. It does not, however, manage to show the data as a whole, and interpreting a data set plotted pairwise can be very overwhelming. Biplots, first introduced by Gabriel (1971), offer a solution to this by providing a method for visualizing a data set with many variables in just two dimensions through approximation.

In biplots, points represent the data set's observations, and non-orthogonal vectors represent variables. Once plotted, one can make various deductions regarding the full data set from the biplot. For example, the more two variables' vectors align with each other, the stronger they are correlated. With similar reasoning, the more similar the values of the two observations on the variables, the closer the points will be to each other in the biplot.

In the simplest of biplots, these approximations make use of singular value decomposition (SVD) and principal component analysis (PCA). Two principal components (usually those associated to the two largest eigenvalues) determine the horizontal and vertical orthogonal axes that provide the coordinates for the approximated observations and variables. These axes are termed the scaffolding axes by Gower and Hand (1996).

Gower and Hand (1996) extended the use of vectors to represent the various variables as calibrated non-orthogonal axes having calibration markers, or simply "biplot axes". These improved biplots more closely resemble scatterplots as it allows for a more straightforward interpretation of the plot. An orthogonal projection from the observations onto the biplot axes provides its immediate approximate predicted value. This process, called visual prediction, implies that one could quickly obtain the estimate of any variable for a specific observation – all in one plot.

One of the problems often encountered when working with these biplots occurs when too many variables are present. Calibrated variable axes' lines (or vectors) and variable labels become crowded and difficult to distinguish from one another. Figure 1 provides an example of such overcrowding within a PCA biplot of the well-known `Iris` data set (Fisher (1936); Anderson (1935)). In this biplot, both the variable names and axis-calibration markers of the two variables `Petal.Width` and `Petal.Length` overlap.

A second problem encountered when working with biplots is a common occurrence with large data sets. In these instances, the data points form a mass, and specific identification of points is no longer possible. It furthermore becomes increasingly difficult to see how dense such a mass of data points are. These masses could hinder the comparison of the density of two clusters or groups of data points; this problem can also be seen in Figure 1, albeit not for such a big data set.

In addition to the shortcomings mentioned above, graphical packages such as base `R` (R Core Team, 2020), which does not allow for live user interactivity of the plot itself, are traditionally used to construct biplots. The use of a graphical package other than base `R` is not a significant problem to correct, but rather, a useful addition that could make biplots even more user-friendly.

The first two problems mentioned above were the main focus of Blasius, Eilers and Gower (2009) in which they proposed methods for improving biplots.

They proposed that one should translate, or shift, the axes parallel relative to their original position to surround the data points rather than intersecting at the origin without any loss of information. These authors, unfortunately, did not provide for translating the axes that would lead to aesthetically pleasing plots. Their solution was to only shift the axes to the edge of the data and then making finer adjustments manually to the plot.

To address the problem of large data sets forming a dense mass of undifferentiated points, they further suggested a type of density plot that would replace the data points. Using a smoothing method, they created a density plot with different colours that indicated how dense the data was over different areas of the biplot.

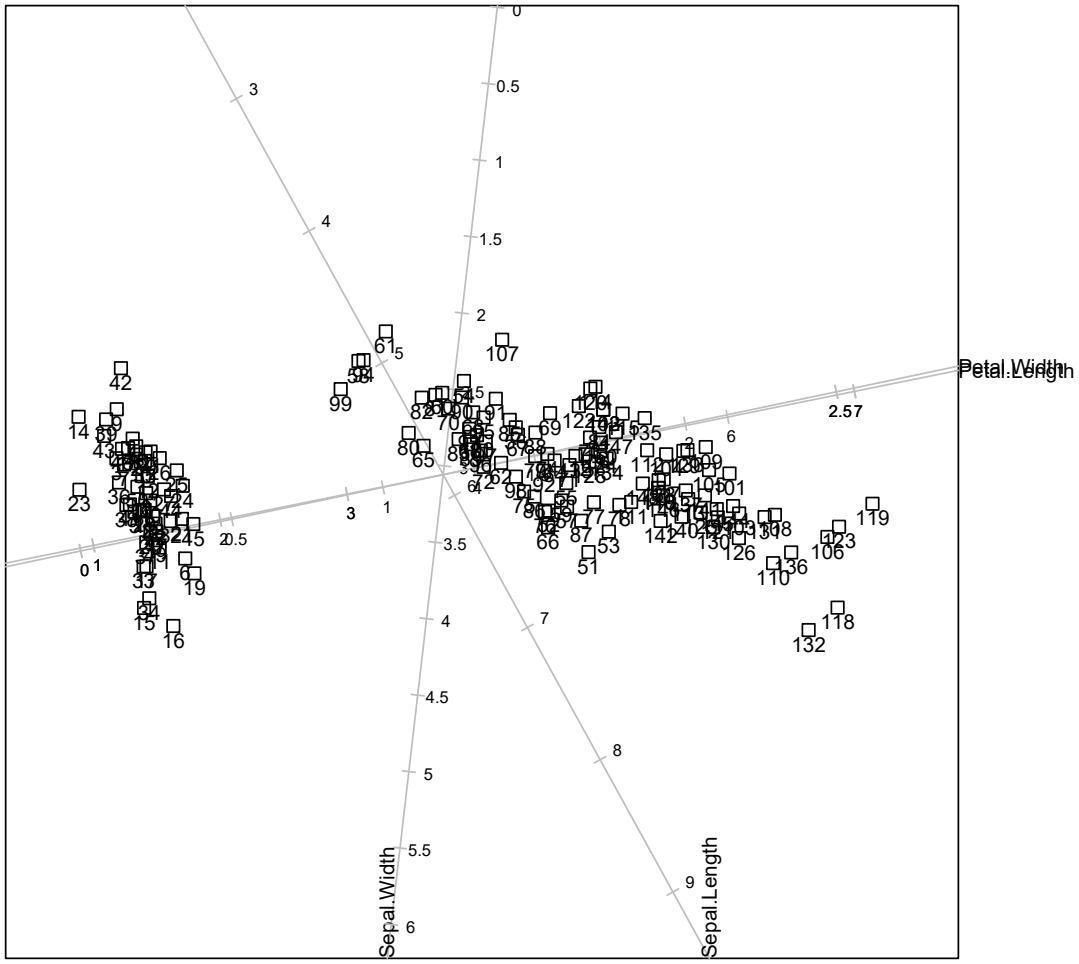


Figure 1: In the above PCA biplot of the well-known `Iris` data set, the variable names, axes, and calibration markers of the two variables `Petal.Width` and `Petal.Length` overlap. This overlap can hinder the accurate interpretation of a biplot. The data furthermore form two groups in which it is difficult to see how the number of data points per group compares to each other, even if one were to give the groups different plotting characters or colours.

In this paper, specific problems encountered when working with biplots, as well as suggestions for making biplots even more functional through the use of user interactivity, are proposed and implemented. An algorithm is proposed to improve the functionalities of biplots through the automatic shifting of the axes and the visualisation of densities on the variable axes. The biplots are also constructed and plotted in `Plotly` (Sievert, 2020), allowing significantly more interactivity for the end-user with the plot itself.

The abovementioned improvements therefore specifically: i) allow for the uncluttering of the biplot space by moving the axes, ii) create densities to summarize crowded data, as well as iii) increase interactivity between the user and the biplot.

This paper is structured as follows: first, a comprehensive overview of available biplot constructing packages are discussed in Section 2. After that, in Section 3, the three main contributions, as mentioned above, are presented. Following this, Section 4 explores the additional minor adjustments to biplots, as well as a discussion of the web-based application created for this paper. The paper is concluded in Section 5.

2. Review of biplot packages

Various packages that can draw biplots exist in `R`, with only a few of them addressing some of the shortcomings discussed in Section 1. A complete list of CRAN packages that contain the term 'biplot' in its name or description helped to identify these. Discussed next are these fourteen packages and how they try to improve biplots.

Various illustrations of these reviewed packages are provided in the Appendix 5 to this paper. All of these uses the well-known `mtcars` (Henderson and Velleman, 1981) data set as the input data. It describes various characteristics of cars (1973 - 1974 models), such as the number of cylinders (`cyl`) or the number of forward gears (`gear`). An adjusted version of the `xclara` data set (Struyf, Hubert, Rousseeuw et al., 1997) is used along with `mtcars` as well. This data set consists of two artificially created variables that neatly group into three equally sized groups. Another variable, with random uniform observations, was added to the data set for illustration purposes. The number of observations in each group was adjusted to show the efficiency of each packages' ability to illustrate different size groups.

Three packages make use of calibrated axes (Gower and Hand, 1996) rather than using vectors – they are `BiplotGUI` (La Grange, Le Roux and Gardner-Lubbe, 2009), `PLSbiplot1` (Oyedele and Gardner-Lubbe, 2014) and `UBbipl`[2] (Gower, Lubbe and Le Roux, 2011).

Of these, only `UBbipl` allows for the translating, or shifting, of the axes. Axes can be moved by either specifying the distance for each separate axis to be moved or selecting a new global intercept point of all the axes. It is, however, essential to note that translating the axes one-by-one could be time-consuming – especially when there are many axes.

[2]While `UBbipl` is not specifically hosted on CRAN, it was still included in this research due to its versatility and its link to the seminal work of Gower et al. (2011).

PLSbiplot1's approach to calibrated axes was different from the others mentioned above, in that it overlays vectors (represented by arrows) on the calibrated axes. While presenting a biplot in this manner could be seen as an improvement, it does, however, overcrowd the biplot even more as the number of variables increases – especially in grayscale.

Four packages address the issue of overcrowding caused by large data sets; these include UBbipl, psych (Revelle, 2019), ade4, and (Dray and Dufour, 2007) . The first three makes use of overlaying density plots where they suggest overlaying the biplot with a density where darker colours indicate a more dense distribution. UBbipl also has the option to plot contour lines. No packages, however, implement the use of a density based on observations' projected predicted values.

Some R packages have a graphical user interface (GUI) capability for plotting biplots. These include BiplotGUI, dynBiplotGUI (Egido, 2017), and ade4TkGUI (Thioulouse and Dray, 2007). While these allow some interactivity between the user and the biplot settings, the interactive functions mostly include changing plotting characteristics such as colours, the changing of biplot types, and including new samples for interpolation. Not one of the fourteen packages, however, made use of the benefits of a live interactive plotting package, such as Plotly – that is, none had the option of interacting with the biplot itself.

Other useful additions to these packages include making use of classes contained in the data. Many packages allow the user to sort the data according to these groups and assign, for instance, different colours, plotting characters, or even draw confidence ellipses or convex hulls around them. Some of these options are available in the following packages: factoextra (Kassambara and Mundt, 2020), psych, ade4, and biplotGUI.

Some other suggestions to make biplots more legible were to move variable labels into legends (BiplotGUI), use a function that will stop any of them from overlapping (factoextra), or allowing for the specifying of own variable text (psych). While these are all useful suggestions, it will not necessarily address the issue of overcrowding in a biplot. If many variables and observations are present, moving the labels to a legend, for example, might not make the biplot easier to interpret.

Six of the abovementioned packages are illustrated in Appendix 5. They include: BiplotGUI, UBbipl, factoextra, PLSbiplot1, psych, and ade4.

A few packages that only draws simplistic biplots considered in the comparison were: stats (R Core Team, 2020) (included in base R), mvdalab (Afanador, Tran, Blanchet and Baumgartner, 2017), sparse (Cubilla-Montilla, Torres, Librero and Villardon, 2019), bpca (Faria, Demétrio and Allaman, 2020), compositions (Van den Boogaart, Tolosana-Delgado and Bren, 2020), dynBiplotGUI (Egido, 2017), and vegan (Oksanen, Blanchet, Friendly, Kindt, Legendre, McGlinn, Minchin, O'Hara, Simpson, Solymos, Stevens, Szoecs and Wagner, 2019). These, however, offered very little plotting freedom and solutions to the problems mentioned earlier and therefore were not included in the illustrations in this paper, nor discussed further.

The code produced for this paper also attempts to improve the visual interpretability and

usefulness of biplots. Illustrations of these enhancements in this paper use the most basic PCA biplots. The contributions discussed next can easily extend to the case of more complex biplots such as Canonical Variate Analysis (CVA) or Analysis of Distance (AOD) biplots. Unless otherwise noted, all biplots use amended code from the `UBbipl` (Gower et al., 2011) package in `R`.

3. Contributions

The shortcomings discussed previously are addressed through i) the automatic translating of the axes to declutter the biplot space, ii) the construction of densities from projected values for better interpretation of groups of data, and iii) the use of `Plotly` to make biplots interactive. These form the basis of this section.

3.1. The automatic translating, or "explosion", of the axes

Blasius et al. (2009) noted that "it is highly improbable that automatic translating of axes will always lead to nice plots". This shortcoming, however, is addressed through an automated algorithm in this paper. The following steps accomplish this:

1. Generate variable vectors, convert to calibrated axes, and trim to include only the range of the projected predicted values for each variable.
2. Determine the coordinates of parallel shifted axes and provide, given their new position, a test whether they intersect with other previously placed axes.
3. Find an iterative process for the automatic outwards translation, or "explosion" of the axes.

These three steps are discussed in detail next.

3.1.1. Creating the biplot axes

It is essential to understand the process of constructing the biplot before considering how they can be enhanced. The process, as proposed by Gabriel (1971), can be described as follows:

Let $\mathbf{X}$ be a scaled and centred data set, and the following be the SVD of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}' \tag{1}$$

where $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values in non-increasing order. $\mathbf{X}$ can then be approximated by

$$\mathbf{X}_r \equiv \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r'$$

where $\mathbf{U}_r$ and $\mathbf{V}_r$ are the first r columns of the matrix $\mathbf{U}$ and $\mathbf{V}$ respectively, and $\mathbf{\Sigma}_r$ is a diagonal matrix containing the r largest singular values.

The two main elements of a biplot, namely the observation points and variable vectors can then be described symbolically as

$$\mathbf{X}_r \equiv (\mathbf{U}_r\mathbf{\Sigma}_r)\mathbf{V}_r' \tag{2}$$

with $\mathbf{X}_r$ being the lower-dimensional approximation to the data matrix $\mathbf{X}$ in terms of two orthogonal vectors, $\mathbf{U}_r\mathbf{\Sigma}_r$ being the observation points to plot, and $\mathbf{V}_r'$ the variable vectors/axes and in the case of a two-dimensional biplot, $r = 2$.

For this paper, the vectors $(\mathbf{V}_r')$ are converted to trimmed biplot axes. These are denoted by their respective start- and end-points, i.e. $\mathbf{a}_i^0$ and $\mathbf{b}_i^0$, for $i = 1, ..., p$, spanned by the scaffolding axes, $\{z_1, z_2\}$. Thus $z_1(\mathbf{a}_i^0)$ denotes the starting point of variable axis i on the z_1 axis and $z_2(\mathbf{a}_i^0)$ the starting point of variable axis i on the z_2 axis.

Similarly, $\{z_1(\mathbf{b}_i^0), z_2(\mathbf{b}_i^0)\}$ is the end-point of axis i. Further note that these will correspond to the maximum and minimum values of the coordinates of the projection for variable i. Figure 2 summarizes this notation that will also be used in the subsequent sections.

3.1.2. Translating and testing for intersecting axes procedure

The process of automatically translating the axes towards the outer edges of the plot consists of trying to find a space for every axis such that not one axis intersects with another. The process followed to obtain the positions of these are given in Algorithm 2. Before this can be applied, however, a method for the parallel translation of the axes must be specified.

The translation of the axes uses the distance between the starting axis and translated axis, together with the gradient. Once shifted parallel to the starting axis (a distance of d) the variable axis $(\mathbf{a}_i^0, \mathbf{b}_i^0)$ will become $(\mathbf{a}_i, \mathbf{b}_i)$. Through considering any point $\mathbf{c}_i^0 = \{z_1(\mathbf{c}_i^0), z_2(\mathbf{c}_i^0)\}$ on the line $(\mathbf{a}_i^0, \mathbf{b}_i^0)$, equations 3, 4, and 5 can be used to obtain $\mathbf{c}_i = \{z_1(\mathbf{c}_i), z_2(\mathbf{c}_i)\}$ as a parallel translation of $\mathbf{c}_i^0$.

Now, given the theoretical distance between points $\mathbf{c}_i^0$ and $\mathbf{c}_i$ is,

$$d = \sqrt{(z_1(\mathbf{c}_i^0) - z_1(\mathbf{c}_i))^2 + (z_2(\mathbf{c}_i^0) - z_2(\mathbf{c}_i))^2}, \tag{3}$$

then, together with the gradient of the line $(\mathbf{a}_i^0, \mathbf{b}_i^0)$, i.e. m_i, the derivation of the new values from the line $(\mathbf{a}_i, \mathbf{b}_i)$ can be found. Since this is a parallel translation, the line $(\mathbf{c}_i^0, \mathbf{c}_i)$ has gradient $-\frac{1}{m_i}$. From this,

$$-\frac{1}{m_i} = \frac{z_2(\mathbf{c}_i) - z_2(\mathbf{c}_i^0)}{z_1(\mathbf{c}_i) - z_1(\mathbf{c}_i^0)}$$

$$-\frac{1}{m_i}(z_1(\mathbf{c}_i) - z_1(\mathbf{c}_i^0)) = z_2(\mathbf{c}_i) - z_2(\mathbf{c}_i^0)$$

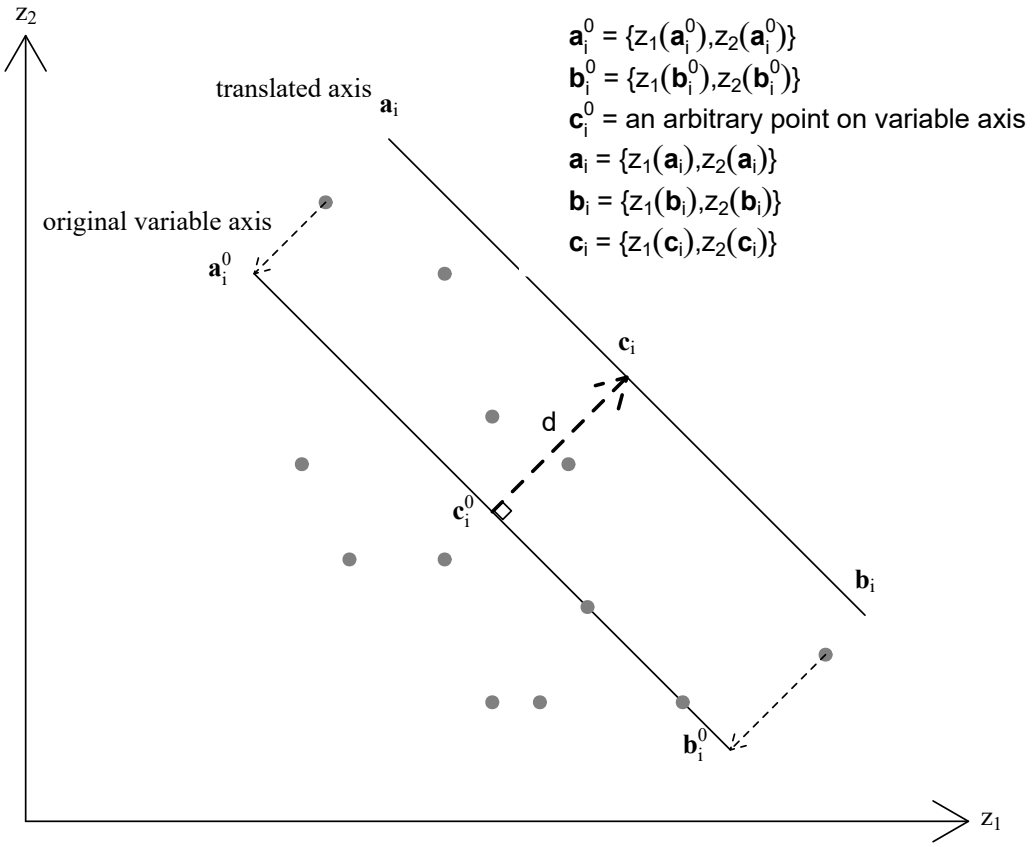


Figure 2: The above diagram shows the notation used in this paper. Each axis i, for $i = 1, ..., p$ with p the number of variables in the data set, can be defined as a line with start- and end-points $(\mathbf{a}_i^0, \mathbf{b}_i^0)$, each having $\{z_1, z_2\}$ coordinates. The translated line, or axis, has been denoted here with a $(\mathbf{a}_i, \mathbf{b}_i)$.

$$z_2(\mathbf{c}_i) = z_2(\mathbf{c}_i^0) - \frac{1}{m_i}(z_1(\mathbf{c}_i) - z_1(\mathbf{c}_i^0)). \tag{4}$$

Substituting the equation of $z_2(\mathbf{c}_i)$ into equation 3, the value of $z_1(\mathbf{c}_i)$ is,

$$\begin{aligned} d &= \sqrt{(z_1(\mathbf{c}_i) - z_1(\mathbf{c}_i^0))^2 + \left(\left(z_2(\mathbf{c}_i^0) - \frac{1}{m_i}(z_1(\mathbf{c}_i) - z_1(\mathbf{c}_i^0))\right) - z_2(\mathbf{c}_i^0)\right)^2} \\ &= (z_1(\mathbf{c}_i) - z_1(\mathbf{c}_i^0))\sqrt{1 + \left(-\frac{1}{m_i}\right)^2} \\ z_1(\mathbf{c}_i) &= z_1(\mathbf{c}_i^0) + \frac{d}{1 + \frac{1}{m_i^2}}, \text{ leading to} \qquad (5) \\ z_2(\mathbf{c}_i) &= z_2(\mathbf{c}_i^0) - \frac{d}{m_i + \frac{1}{m_i}}. \qquad (6) \end{aligned}$$

Therefore, the coordinate pair of the translated position of $\{z_1(\mathbf{c}_i^0), z_2(\mathbf{c}_i^0)\}$, namely $\{z_1(\mathbf{c}_i), z_2(\mathbf{c}_i)\}$, is calculated using equation 5 and 6.

Because axes translate in a parallel fashion, the calibration of the axes does not need to change. Note that these axes can move in a positive or negative direction by setting $d = -d$, and the distance can be increased or decreased by adding a small value $\delta_d > 0$ to $|d|$. These input values and necessary equations are used in the automated algorithm for translating the axes, discussed next.

The translation of axes towards the edges of the biplot is central to improving their usefulness. To find a position on the plot where each axis will have sufficient space, one needs to be able to test whether axes intersect with one another. These intersections can be the case where they intersect at a specific point, or in the case of parallel axes, where they intersect at multiple locations.

Now, given an initial axis, $(\mathbf{a}_i^0, \mathbf{b}_i^0)$, and its first translation $(\mathbf{a}_i, \mathbf{b}_i)$, to test whether it intersects with another previously translated axis $(\mathbf{a}_j, \mathbf{b}_j)$, for $1 \leq j < (i-1)$ and $i = 1, ..., p$, Algorithm 1 was applied. This procedure was suggested by Cormen, Leiserson, Rivest and Stein (2009) for testing whether lines intersect or overlap each other. The process makes use of calculating the orientation of the lines as seen from a fixed point (an end-point of one of the lines).

The following equations, written in terms of three generic points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ in the $\{z_1, z_2\}$ plane, are required to apply the above. Here $g(f(\mathbf{a}, \mathbf{b}, \mathbf{c}))$'s output provides information on the direction of the three points, and $h(\mathbf{a}, \mathbf{b}, \mathbf{c})$ together with $g(f(\mathbf{a}, \mathbf{b}, \mathbf{c}))$ checks whether they are co-linear.

$$\begin{aligned} f(\mathbf{a}, \mathbf{b}, \mathbf{c}) &= (\mathbf{c} - \mathbf{a}) \times (\mathbf{b} - \mathbf{a}) \\ &= (z_1(\mathbf{a}) - z_1(\mathbf{c}))(z_2(\mathbf{a}) - z_2(\mathbf{c})) - (z_1(\mathbf{b}) - z_1(\mathbf{a}))(z_2(\mathbf{b}) - z_2(\mathbf{a})) \end{aligned} \tag{7}$$

$$g(X) = \begin{cases} 1 & X > 0 \\ 2 & X < 0 \\ 0 & \text{else} \end{cases} \tag{8}$$

$$h(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{cases} T & \min(z_1(\mathbf{a}), z_1(\mathbf{b})) \leq z_1(\mathbf{c}) \leq \max(z_1(\mathbf{a}), z_1(\mathbf{b})) \ \cap \\ & \min(z_2(\mathbf{a}), z_2(\mathbf{b})) \leq z_2(\mathbf{c}) \leq \max(z_2(\mathbf{a}), z_2(\mathbf{b})) \\ F & \text{otherwise} \end{cases} \tag{9}$$

These formulae are used in Algorithm 1 to determine whether two line segments intersect.

Algorithm 1: Test if two lines intersect each other

Input: lines i and j represented by their respective beginning and end-points $(\mathbf{a}_i, \mathbf{b}_i)$ and $(\mathbf{a}_j, \mathbf{b}_j)$;
$T_1 = g(f(\mathbf{a}_j, \mathbf{b}_j, \mathbf{a}_i))$;
$T_2 = g(f(\mathbf{a}_j, \mathbf{b}_j, \mathbf{b}_i))$;
$T_3 = g(f(\mathbf{a}_i, \mathbf{b}_i, \mathbf{a}_j))$;
$T_4 = g(f(\mathbf{a}_i, \mathbf{b}_i, \mathbf{b}_j))$;
if $T_1 \neq T_2 \cap T_3 \neq T_4$ **then** TRUE;
if $T_1 = 0 \ \cap \ h(\mathbf{a}_j, \mathbf{b}_j, \mathbf{a}_i) = T$ **then** TRUE;
if $T_2 = 0 \ \cap \ h(\mathbf{a}_j, \mathbf{b}_j, \mathbf{b}_i) = T$ **then** TRUE;
if $T_3 = 0 \ \cap \ h(\mathbf{a}_i, \mathbf{b}_i, \mathbf{a}_j) = T$ **then** TRUE;
if $T_4 = 0 \ \cap \ h(\mathbf{a}_i, \mathbf{b}_i, \mathbf{b}_j) = T$ **then** TRUE;
else FALSE;

3.1.3. Automated algorithm to translate the axes

Finally, a discussion of the process for automating the parallel translation of axes follows. Firstly, one needs to find D_{base}, the global minimum distance to move each axis. This distance is the starting value chosen to test whether an axis needs to shift by D_{base} or further. In this paper, this value is equal to the radius of the smallest possible circle around the data, or any other variation that includes only a specific portion of the data (possibly excluding outliers). In the code produced for this paper, D_{base} has been chosen as a fixed value but can be dynamic as well. Setting D_{base} to a fixed value results in a more symmetric plot, which generally gives more aesthetically pleasing results rather than using a variable D_{base} based on the distance from, for example, the centroid of an ellipse.

Once D_{base} has been set, the algorithm tests whether this distance will ensure no axis intersects with another. It has to compare each newly translated axis with all previous axes. If a newly translated axis intersects with a previously translated axis, then it is first mirrored

by multiplying D_{base} with -1. If it still intersects, then a small distance δ_D is added. This process is repeated until the translated axis does not intersect with any previously translated axes.

Algorithm 2 provides a summary of the process on how this is done and an illustration of moving the axes can be seen in Figure 3 using the `R` program written for this paper.

Algorithm 2: Automatic translation of the axes

D_{base} := minimum distance of a parallel translation of an axis from the origin;
$mult$:= multiplication factor to indicate direction;
$m_i := \frac{z_2(\mathbf{b}_i)-z_2(\mathbf{a}_i)}{z_1(\mathbf{b}_i)-z_1(\mathbf{a}_i)}$ calculated from the start- ($\mathbf{a}_i$) and end-points ($\mathbf{b}_i$) of axis i;
δ_D := a small distance used for incremental shifting;
for $i \leftarrow 1$ **to** p **do**
  $mult = (-1)^{i+1}$;
  $D_{shift} = D_{base}$;
  $z_1(\mathbf{a}_i, \mathbf{b}_i) = z_1(\mathbf{a}_i^0, \mathbf{b}_i^0) + \frac{D_{shift} \times mult}{1+1/m_i^2}$;
  $z_2(\mathbf{a}_i, \mathbf{b}_i) = z_2(\mathbf{a}_i^0, \mathbf{b}_i^0) - \frac{D_{shift} \times mult}{m_i+1/m_i}$;
  if $i > 1$ **then**
    for $j \leftarrow 1$ **to** $i-1$ **do**
      $t = 0$;
      while *Algorithm 1 = TRUE for* $\{(\mathbf{a}_i, \mathbf{b}_i), (\mathbf{a}_j, \mathbf{b}_j)\}$ **do**
        $t \leftarrow t + 1$;
        $mult \leftarrow (-1)^{i+t}$;
        $D_{shift} \leftarrow D_{base} + \delta_D \times \lfloor \frac{t+1}{2} \rfloor$;
        $z_1(\mathbf{a}_i, \mathbf{b}_i) \leftarrow z_1(\mathbf{a}_i^0, \mathbf{b}_i^0) + \frac{D_{shift} \times mult}{1+1/m_i^2}$;
        $z_2(\mathbf{a}_i, \mathbf{b}_i) \leftarrow z_2(\mathbf{a}_i^0, \mathbf{b}_i^0) - \frac{D_{shift} \times mult}{m_i+1/m_i}$;
    next j
next i

3.2. Creating densities on the axes

A variation of orthogonal projection, as proposed by Van der Merwe (2020), was used to address the problem of overcrowding in a biplot with a large number of observations. Through a process of projection onto axes, one can illustrate the different groups of data points to specific classification regions. Summarizing the data in densities on each axis could serve as an alternative to observing the detail of every data point. These density summaries are obtained by projecting predicted values of each point on each axis, which were in turn used to construct the density. The densities of each projected variable are a good representation of the observations on the biplot. With this, if the observations are almost entirely indistinguishable, one could remove the data points and simply make use of the densities, without loss of too much information.

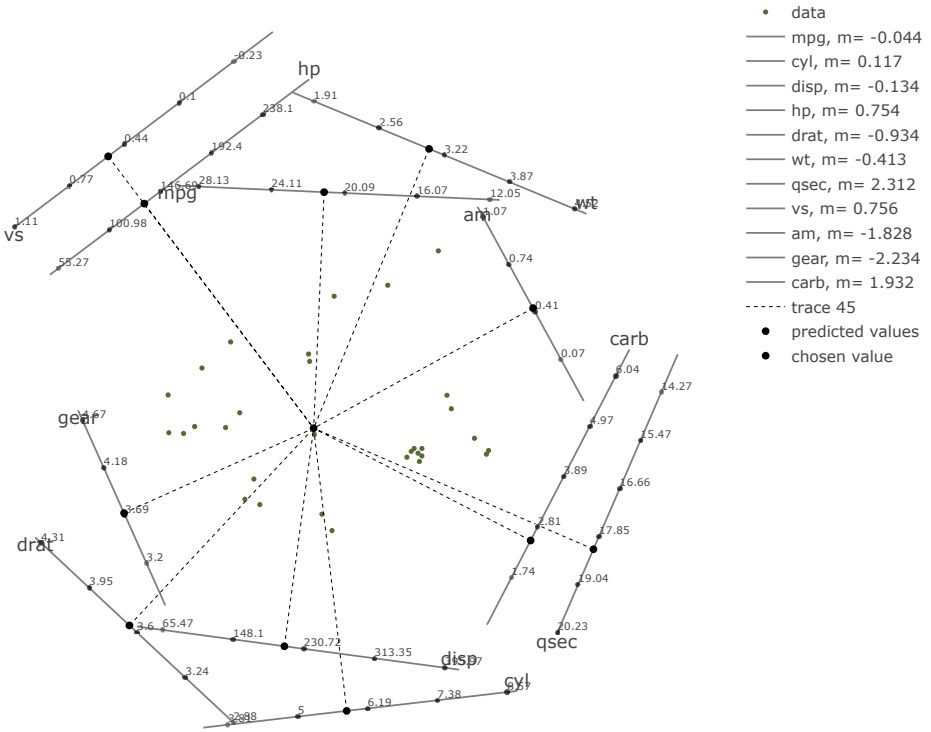


Figure 3: A PCA biplot of the `mtcars` data set after translating the axes according to the Algorithm 2. By moving them towards the edge of the plotted data set, it results in a more legible, interpretable, and aesthetically pleasing biplot. An arbitrary observation's predicted values are also visible on the biplot. An HTML version of this plot is available at: https://doi.org/10.5281/zenodo.4066399. The plot has various interactive capabilities, such as observing the sample names when hovering over the data points and adding and removing elements of the plot.

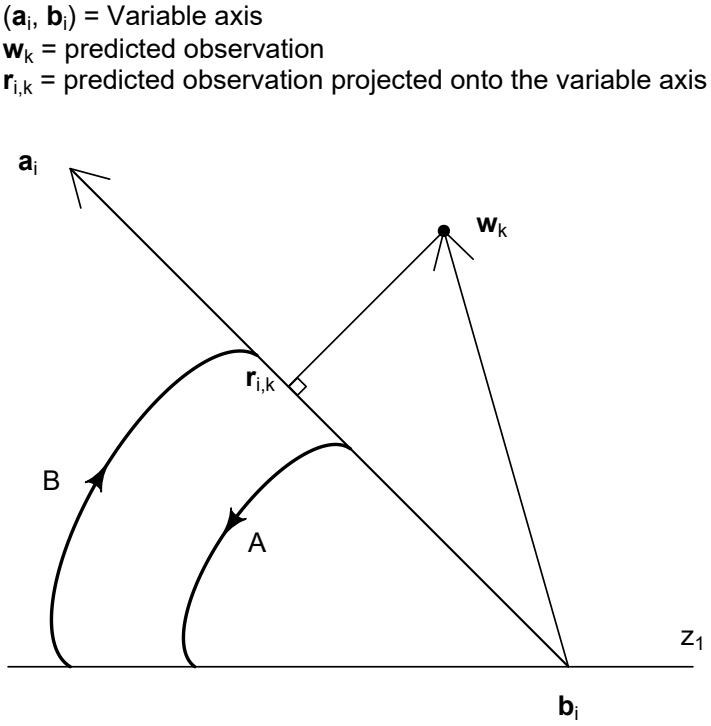


Figure 4: The above diagram illustrates the notation for deriving densities on the variable axes. This process consists of considering a predicted observation $\mathbf{w}_k$ that is projected onto the variable axis $\mathbf{a}_i, \mathbf{b}_i$. This is followed by process A that describes rotating the projected observations points onto the z_1 axis. At this point a histogram is constructed and smoothed, which is then rotated back (B) onto the variable axis. This is explained in Algorithm 3

Figure 4 provides a summary of the notation used in the derivation. Here the observations $\mathbf{w}_k, k = 1, ..., N$ are projected onto the variable axes $(\mathbf{a}_i, \mathbf{b}_i)$, whereafter they are rotated onto the z_1 axis. A histogram is then constructed on the z_1 axis and smoothed to form a density like structure that is rotated back to the position of the variable axis. The rotation matrix used is

$$T(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta, \end{bmatrix}, \tag{10}$$

where θ is the angle corresponding to the gradient of the different variable axes $(tan^{-1}(m_i))$. The process for creating these densities is discussed in Algorithm 3.

In Algorithm 3, $\mathbf{r}_i$ represents the projected observations onto the axes. These are rotated to $\mathbf{r}'_i$ on the z_1 scaffolding axis, and a density is constructed through the use of midpoints represented by the sequence $\{Q_{i,c}\}$, with c the number of midpoints. The subsequent density, $\mathbf{h}_i$, is then rotated back onto the original axis to form $\mathbf{h}'_i$. All of these are spanned in z_1 and z_2.

Algorithm 3: Creation of densities projected on biplot axes

$\mathbf{s}_i := (\mathbf{a}_i, \mathbf{b}_i)$ = biplot axis for variable i in $i = 1, ..., p$;
$m_i :=$ gradient of the variable axis $\mathbf{s}_i$;
$\mathbf{w}_k := \{z_1(\mathbf{w}_k), z_2(\mathbf{w}_k)\}$ coordinates of an observation $\mathbf{w}_k$ for $k = 1, ..., N$ in the biplot space $\{z_1, z_2\}$;
$\mathbf{r}_{i,k} := \{z_1, z_2\}$ coordinates of the the projected values of the k^{th} observation onto the i^{th} biplot axis;
$\mathbf{r}'_{i,k} :=$ values of $\mathbf{r}_{i,k}$ rotated onto the z_1 axis;
$\mathbf{r}'_i :=$ vector of all the $z_1(\mathbf{r}'_{i,k})$ for $k = 1, ..., N$ for variable i;
$R :=$ the number of bins $= round(\sqrt{N})$;
$b := \frac{\max(z_1(\mathbf{r}'_i)) - \min(z_1(\mathbf{r}'_i))}{R}$;
$\{Q_{i,c}\}_{c=1,...,R} := \{\min(z_1(\mathbf{r}'_i)) + b/2 + (c-1) \times b\}$;
$\mathbf{h}_i :=$ matrix of $\{z_1, z_2\}$ of the column heights of a histogram;
$\mathbf{h}'_i :=$ rotated $\{z_1, z_2\}$ coordinates of $\mathbf{h}_i$;
for $i \leftarrow 1$ **to** p **do**
  for $k \leftarrow 1$ **to** N **do**
    $\mathbf{r}_{i,k} = \frac{\boldsymbol{w}_k \cdot \mathbf{s}_i}{\mathbf{s}_i \cdot \mathbf{s}_i}$;
    $\mathbf{r}'_{i,k} = T(-\tan^{-1}(m_i)) \times \mathbf{r}_{i,k}$;
  next k;
  Construct an histogram, anchored on z_1, with $\{Q_{i,c}\}_{c=1,...,R}$ as midpoints delivering $\mathbf{h}_i$;
  Apply a smoothing spline to $\mathbf{h}_i$;
  $\mathbf{h}'_i = T(\tan^{-1}(m_i)) \times \mathbf{h}_i$;
next i

These density lines are smoothed histogram heights calculated with a smoothing spline from the function `smooth.spline` available in the `stats` package. Note further that a manually adjustable smoothing parameter controls this smoothing process, which allows for the control between overfitting and underfitting by the user using the parameter `smoothing_par`. Setting the smoothing setting equal to NULL provides an automatically calculated smoothing parameter in the function `smooth.spline()`. This value is typically between zero and one where the larger the number, the more smoothing occurs.

One could further expand the abovementioned densities to different pre-specified groups of data, which would lead to multiple density plots on each axis. In the code produced for this paper, the `cluster_var` parameter will initiate this option to create separate densities for separate groups, illustrated by Figure 5. In this figure, there are three variables and three groups, each with their density, showing that all three groups have differing amounts of observations.

The densities supply further additional information not readily available in a normal biplot, such as the ones discussed in the packages in Section 2. For example, the observations in group three are more condensed for variable one, compared to group one. Also, in the midpoint of the two groups, there are more values in the third than in the first. All this information can now easily be interpreted directly from the biplot.

3.3. Interactive biplots using `Plotly`

The use of `Plotly`, a plotting package available in `R`, adds value to biplots in that the user can engage interactively with the plot itself. While other biplot packages with GUIs allow for some interactivity between the user and the biplot, the interactive functions mostly include changing characteristics such as colours of plotting characters, the changing of biplot types, and including new samples for interpolation. Some allow the inserting of confidence ellipses or convex hulls as well as grouping functions. These are all very useful, but the authors have yet to find plotting packages allowing the level of interactive freedom as seen in `Plotly`.

One of the most significant benefits of using `Plotly` is its interactive capabilities to display values on hover. This capability allows users to see variable or observation names and coordinates. It also allows the user to remove labels to make the plot more legible while still conveying the identification of variables or observations.

Another interactive function is that one can remove axes. Visualizing only specific axes is ideal for comparing specific variables (axes) to one another. By also being able to remove variables or groups of sample points, it allows one to focus on particular parts of the biplot. These, however, still form part of the construction of the biplot. It also allows zooming and selecting certain areas to inspect. In addition to this, `Plotly` is also able to export interactive plots as HTML files. Having standalone files implies that the user can interact with biplots, effectively bypassing the use of `R` for the end-user.

A `Shiny` web-based GUI that can produce these downloadable HTML images by `Plotly`, together with some additional minor improvements, are discussed in the following section.

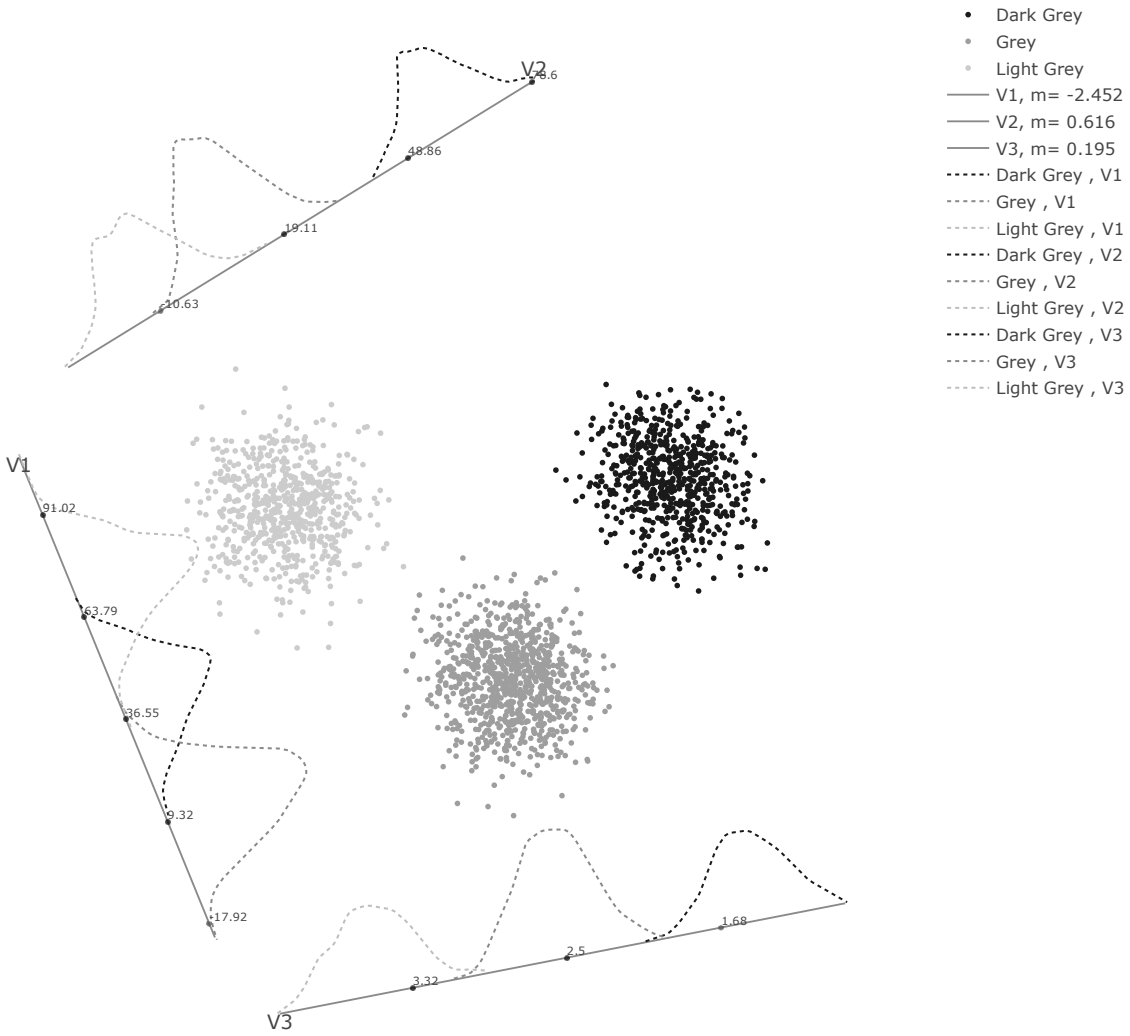


Figure 5: The above biplot of the adapted `xclara` data set shows three grouped densities projected onto its three variables' axes. The biplot illustrates its clear benefit in that more information about the groups is available. The light grey group seems more concentrated than the dark grey one, and there appear to be more data points in the former than the latter. An HTML version of this plot is available at: https://doi.org/10.5281/zenodo.4066399.

4. Further minor adjustments and `Shiny` application

Given the already discussed primary adjustments, this section elaborates on the minor, secondary, enhancements added to the code developed for this paper. The reader can interact with the web-based `Shiny` GUI application online at https://carelvdmerwe.shinyapps.io/ExplodingBiplots/. This application makes use of various data sets to illustrate the different aspects discussed. These data sets include `Iris`, the adjusted `xclara`, as well as the `mtcars` data set which have been mentioned previously. Two additional data sets have also been used for illustration purposes, these include the `Ocotea` (Swart, 1985) and `Copperfroth` (Aldrich et al., 2004) data sets. The `Ocotea` data set describes the anatomical characteristics of wood having observations from three distinct classification categories, while the `Copperfroth` data set contains the characteristics of the appearance of copper froth from a copper flotation plant in South Africa identified by five different froth structures.

All of the secondary adjustments mentioned below have been illustrated in the `Shiny` application as well which allows one to download an HTML file of the plots similarly to `Plotly`.

The secondary enhancements include functionalities surrounding: i) the calibration markers, ii) the projected predicted values, iii) density plots, and iv) dealing with outliers.

4.1. Calibration markers

One of the adjustments incorporated into the `R` program relates to calibration markers. The construction of these markers, specifically on a prediction biplot, uses the methodology of Gower and Hand (1996), which comprised of finding the length of one unit of a specific variable axis and can be calculated using equation 11.

$$\frac{\boldsymbol{V}_r^T \boldsymbol{e_u}}{\boldsymbol{e}_u^T \boldsymbol{V}_r \boldsymbol{V}_r^T \boldsymbol{e_u}} \tag{11}$$

As described by Blasius et al. (2009) it follows that in the above equation, $\boldsymbol{e}_u$ is a unit vector with the value of one in the k^{th} position, whilst $\mathbf{V}$ is the matrix as calculated in equation 2. Equation 11 produces a $\{z_1, z_2\}$ coordinate, providing the change in z_1 or z_2 for one unit change in the value of the variable. A sequence of $\{z_1, z_2\}$ coordinates is then created with corresponding calibration markers. The markers shown on the plot are those that lie within the range of the projected predicted values for a specific variable. By using the projected predicted values' range, it allows the user to inspect even the most extreme points through visual projection or the prediction functionality in the code. The calibrations on the axes can furthermore be adjusted to display more tick-marks.

The final changes made to the calibration markers include flexibility in the plotting size, rounding to the desired amount of decimals, and setting the scale of the markers to be either standardized or the same as the original data set.

4.2. Projected predicted values

Another secondary adjustment, incorporated based on the work of Gower and Hand (1996), was to show the projected predicted values of a specific observation for each variable on the biplot itself. In the `R` program developed for this paper one can find specific projected predicted values on the plot by choosing it by name or by the row number of the observation. The function will produce the projected predicted values on each axis with interactive labels displaying the values on hover.

4.3. Density plots

The density plots also introduce some flexibility. If there are too few data points to construct a meaningful density, one can choose not to plot the densities. There is also an added functionality to inflate (or deflate) the densities by a predetermined factor. Changing the size of the density curves would be necessary if some overlapped with other axes lines or overcrowded the plot. In this instance, the user would be able to deflate the densities to make them fit in the allocated space. This solution is feasible since the main aim of the density curves are the shape and how they compare to one another, not their actual values. Another solution to the densities and axes that overlap would be to adjust the parameter δ_D. Increasing this parameter would, in most cases, create larger spaces between the axes allowing for more space for the densities.

4.4. Dealing with outliers in the data

Lastly, a potential issue that could arise from using Algorithm 2 is that when data has outliers, the smallest possible circle surrounding the data could be much larger than the majority of the points. Consequently, the axes would be translated to a position far away from the bulk of the data. The `R` program developed has the option of selecting an ellipse that only includes a certain percentile of the data, for example, 90%, to address this. Using a smaller ellipse would allow outliers to still feature on the biplot without affecting the positioning of the axes on the biplot.

5. Conclusion

Biplots are extremely useful in visualizing multidimensional data. As such, the focus of this paper was on the various options to enhance their ability to illustrate the data more effectively. A whole new interactive sphere of biplots that uses `Plotly`, in which one can dissect a biplot layer by layer into its components, has been introduced. This interactive capability is also useful when comparing certain variables to each other by removing unnecessary elements. The automatic translating of axes, furthermore, opens up the centre of the biplot to not let labels and data points overlap. This shifting of the axes enables the user to see calibration markers clearly and to interpret biplots more accurately. Finally, creating densities allows one to see how dense a group of data points is. This information could get

lost in a simplistic biplot. These adjustments make biplots more powerful when having to interpret and analyse them.

References

Afanador, N.L., Tran, T., Blanchet, L., Baumgartner, R., 2017. mvdalab: Multivariate Data Analysis Laboratory. URL: https://CRAN.R-project.org/package=mvdalab. R package version 1.4.

Aldrich, C., Gardner, S., Le Roux, N., 2004. Monitoring of metallurgical process plants by using biplots. AIChE Journal 50, 2167–2186. URL: http://doi.org/10.1002/aic.10170, doi:10.1002/aic.10170.

Anderson, E., 1935. The irises of the gaspe peninsula. Bull. Am. Iris Soc. 59, 2–5.

Blasius, J., Eilers, P.H., Gower, J., 2009. Better biplots. Computational Statistics & Data Analysis 53, 3145–3158. URL: http://doi.org/10.1016/j.csda.2008.06.013, doi:10.1016/j.csda.2008.06.013.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to algorithms. MIT press.

Cubilla-Montilla, M., Torres, C., Librero, A.B.N., Villardon, P.G., 2019. SparseBiplots: 'HJ Biplot' using Different Ways of Penalization. URL: https://CRAN.R-project.org/package=SparseBiplots. R package version 3.5.0.

Dray, S., Dufour, A.B., 2007. The ade4 package: Implementing the duality diagram for ecologists. Journal of Statistical Software 22, 1–20. URL: http://doi.org/10.18637/jss.v022.i04, doi:10.18637/jss.v022.i04.

Egido, J., 2017. dynBiplotGUI: Full Interactive GUI for Dynamic Biplot in R. URL: https://CRAN.R-project.org/package=dynBiplotGUI. R package version 1.1.5.

Faria, J.C., Demétrio, C.G.B., Allaman, I.B., 2020. bpca: Biplot of Multivariate Data Based on Principal Components Analysis. UESC and ESALQ. Ilheus, Bahia, Brasil and Piracicaba, Sao Paulo, Brasil. URL: https://cran.r-project.org/web/packages/bpca/.

Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Annals of eugenics 7, 179–188. URL: http://doi.org/10.1111/j.1469-1809.1936.tb02137.x, doi:10.1111/j.1469-1809.1936.tb02137.x.

Gabriel, K.R., 1971. The biplot graphic display of matrices with application to principal component analysis. Biometrika 58, 453–467. URL: http://doi.org/10.2307/2334381, doi:10.2307/2334381.

Gower, J., Hand, D., 1996. Biplots. Chapman & Hall.

Gower, J.C., Lubbe, S.G., Le Roux, N.J., 2011. Understanding biplots. John Wiley & Sons.

Henderson, H.V., Velleman, P.F., 1981. Building multiple regression models interactively. Biometrics , 391–411URL: http://doi.org/10.2307/2530428, doi:10.2307/2530428.

Kassambara, A., Mundt, F., 2020. factoextra: Extract and Visualize the Results of Multivariate Data Analyses. URL: https://CRAN.R-project.org/package=factoextra. R package version 1.0.7.

La Grange, A., Le Roux, N., Gardner-Lubbe, S., 2009. BiplotGUI: Interactive biplots in R. Journal of Statistical Software 30, 1–37. URL: http://doi.org/10.18637/jss.v030.i12, doi:10.18637/jss.v030.i12.

Oksanen, J., Blanchet, F.G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P.R., O'Hara, R.B., Simpson, G.L., Solymos, P., Stevens, M.H.H., Szoecs, E., Wagner, H., 2019. vegan: Community Ecology Package. URL: https://CRAN.R-project.org/package=vegan. R package version 2.5-6.

Oyedele, O.F., Gardner-Lubbe, S., 2014. PLSbiplot1: The Partial Least Squares (PLS) Biplot. URL: https://CRAN.R-project.org/package=PLSbiplot1. R package version 0.1.

R Core Team, 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Revelle, W., 2019. psych: Procedures for Psychological, Psychometric, and Personality Research. Northwestern University. Evanston, Illinois. URL: https://CRAN.R-project.org/package=psych. R package version 1.9.12.

Sievert, C., 2020. Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/CRC. URL: https://plotly-r.com.

Struyf, A., Hubert, M., Rousseeuw, P., et al., 1997. Clustering in an object-oriented environment. Journal of Statistical Software 1, 1–30. URL: http://doi.org/10.18637/jss.v001.i04, doi:10.18637/jss.v001.i04.

Swart, J., 1985. 'n Sistematies-houtanatomiese ondersoek van die Lauraceae in Suidelike Afrika. Master's thesis. Stellenbosch: Stellenbosch University.

Thioulouse, J., Dray, S., 2007. Interactive multivariate data analysis in R with the ade4 and ade4tkgui packages. Journal of Statistical Software 22, 1–14. URL: http://doi.org/10.18637/jss.v022.i05, doi:10.18637/jss.v022.i05.

Van den Boogaart, K.G., Tolosana-Delgado, R., Bren, M., 2020. compositions: Compositional Data Analysis. URL: https://CRAN.R-project.org/package=compositions. R package version 1.40-5.

Van der Merwe, C.J., 2020. Classifying yield spread movements in sparse data through triplots. Ph.D. thesis. Ghent University & Stellenbosch University. URL: http://hdl.handle.net/1854/LU-8643411.

Appendix A Biplot Package Comparison

This supplementary material to "Exploding biplots with density axes in `Plotly`" contains all the figures produced for section "Review of biplot packages".

The biplots that follow are drawn from the `mtcars` and adapted `xclara` data sets. The code for all of these can be found in the `OtherBiplots.R` supplementary file.

The following packages are illustrated below in Figures A.1 to A.6:

Figure A.1: `BiplotGUI` (https://cran.r-project.org/web/packages/BiplotGUI/index.html)

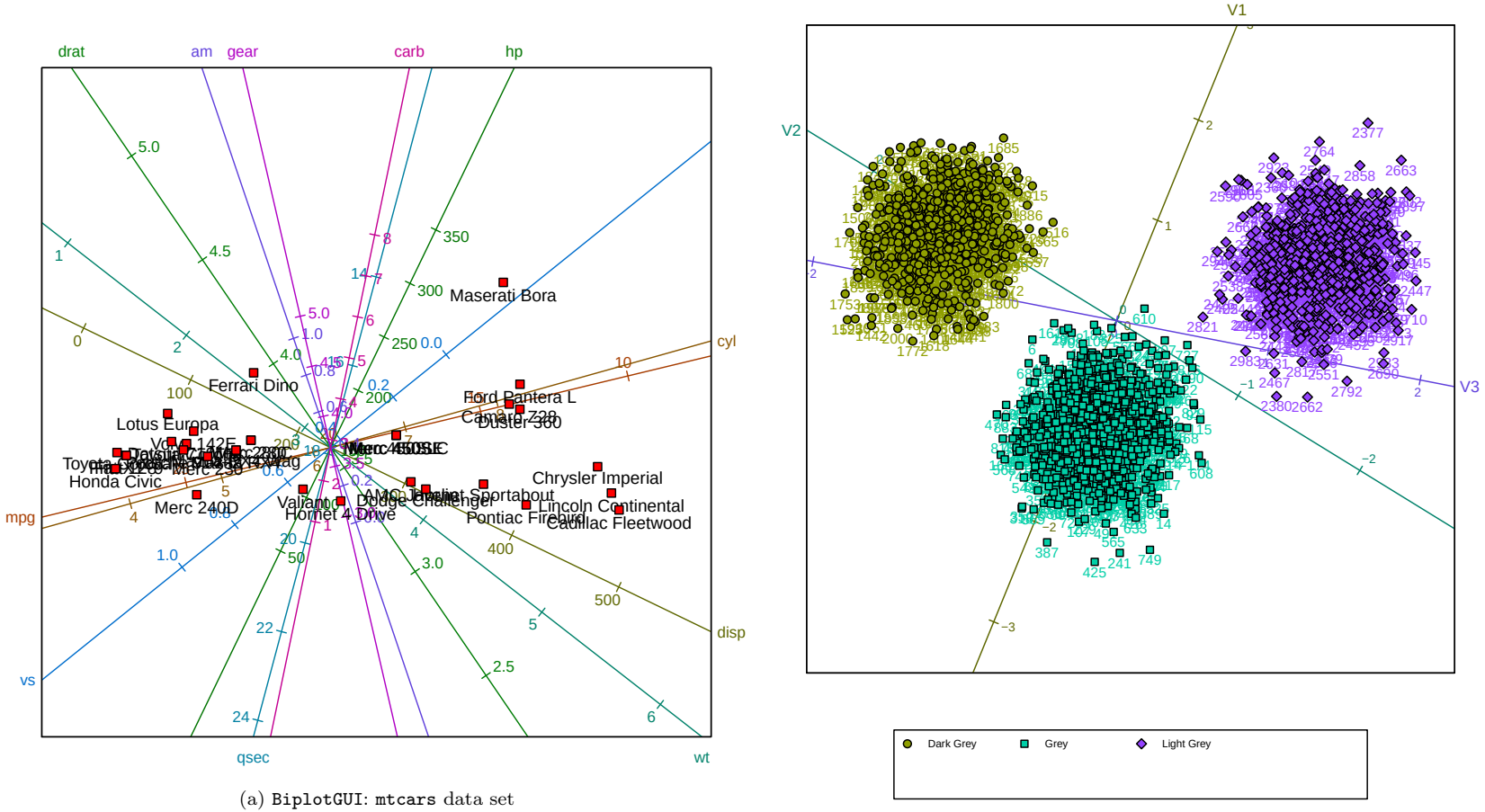
Figure A.2: `UBbipl` (https://www.wiley.com/legacy/wileychi/gower/material.html)

Figure A.3: `factoextra` (https://cran.r-project.org/web/packages/factoextra/index.html)

Figure A.4: `PLSbiplot1` (https://cran.r-project.org/web/packages/PLSbiplot1/index.html)

Figure A.5: `psych` (https://cran.r-project.org/web/packages/psych/index.html)

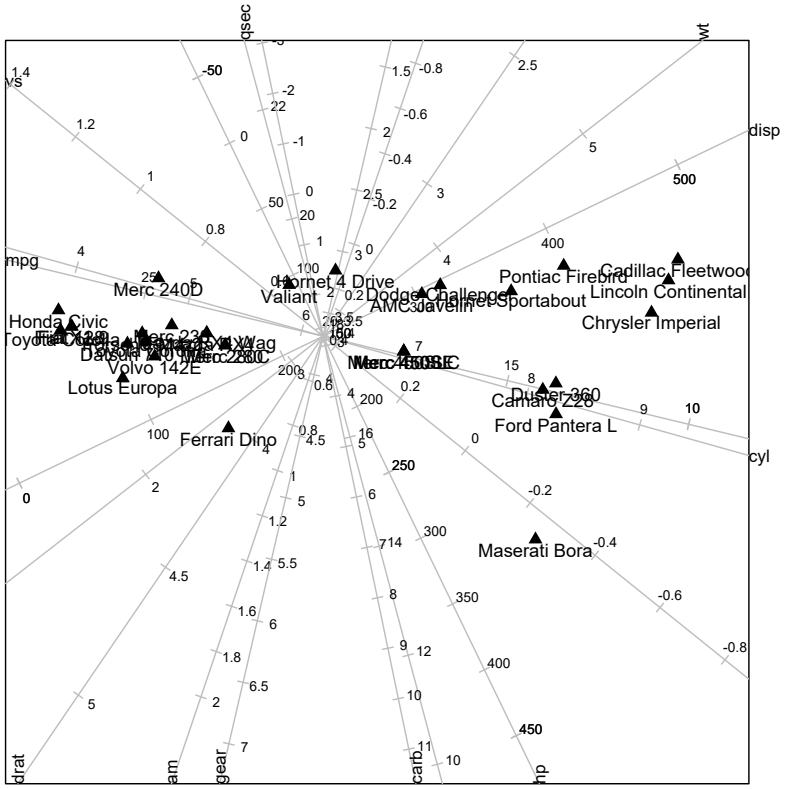
Figure A.6: `ade4` (https://cran.r-project.org/web/packages/ade4/index.html)



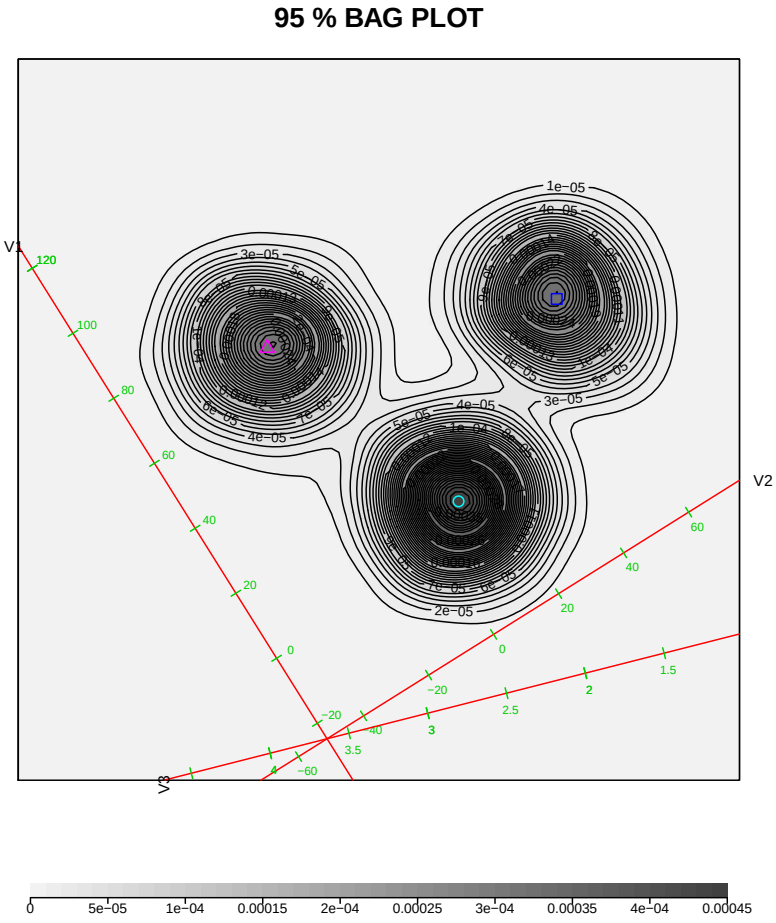
(a) `BiplotGUI`: `mtcars` data set

(b) `BiplotGUI`: `xclara` data set

Figure A.1: `BiplotGUI`, depicted above, illustrating a biplot with calibrated axes. In A.1(a), one can see the overlapping of variables and observations. The package allows one to remove the data point labels and move the axis labels to a legend at the side of the plot to address this. In the case of A.1(b), the only solution suggested for a mass of data is by using point densities. These densities are colour shadings showing one how dense it is at a certain point. The calibration of axes is useful but can often add to the busyness of the biplot, making inference more difficult. This package furthermore allows for various other advanced biplot functionalities not illustrated here.

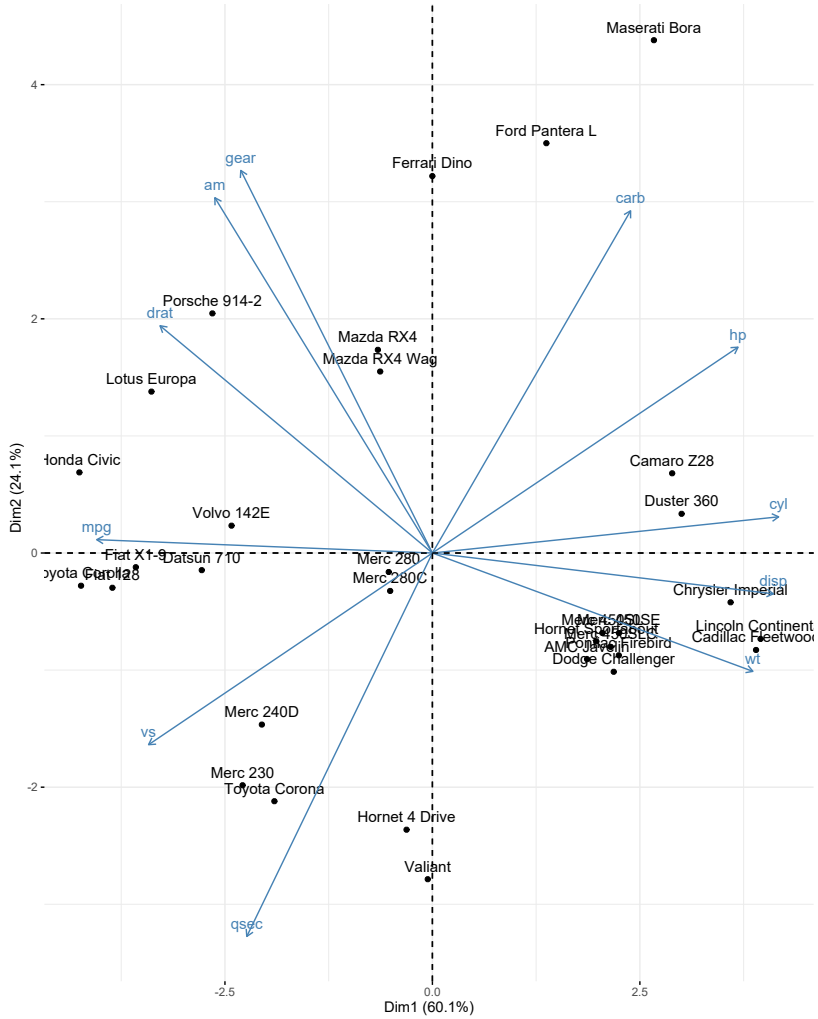


(a) `UBbipl`: `mtcars` data set

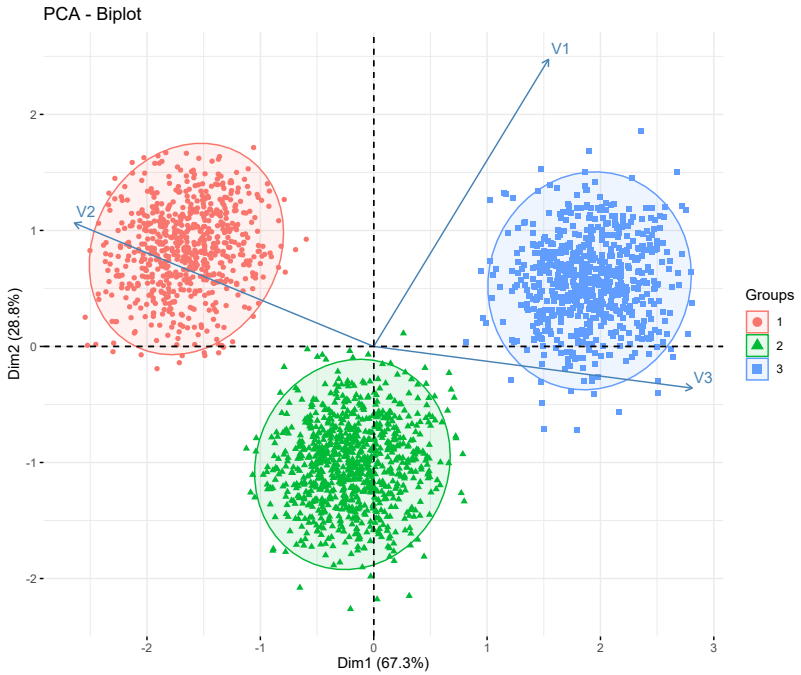


(b) `UBbipl`: `xclara` data set

Figure A.2: `UBbipl` offers much flexibility in its plotting capabilities. There are multiple parameter options, such as colour schemes and plotting characters. Along with all this flexibility, there are many adjustments to the labels and axes markers available, which include size and location adjustments as well as the ability to zoom in or out of the biplot. One can also remove the labels and adjust the axes markers to show only extremes, which should unclutter the biplot. Unfortunately, as can be seen in A.2(a), the axes and points still overlap, and at the midpoint where all the axes meet, it looks quite overwhelming at first glance. The proposed solution, given in A.2(b), is to move axes manually or interactively choosing a new origin. It is important to note that moving the axes one by one and finding the optimal positions can be time-consuming. The only solutions suggested when data groups are present would be the adjustment of labels and plotting characters above. There are, however, various options for drawing ellipses, convex hulls, and α-bags around certain groups which could help differentiate groups, but not necessarily comment on how dense each of them is in detail.

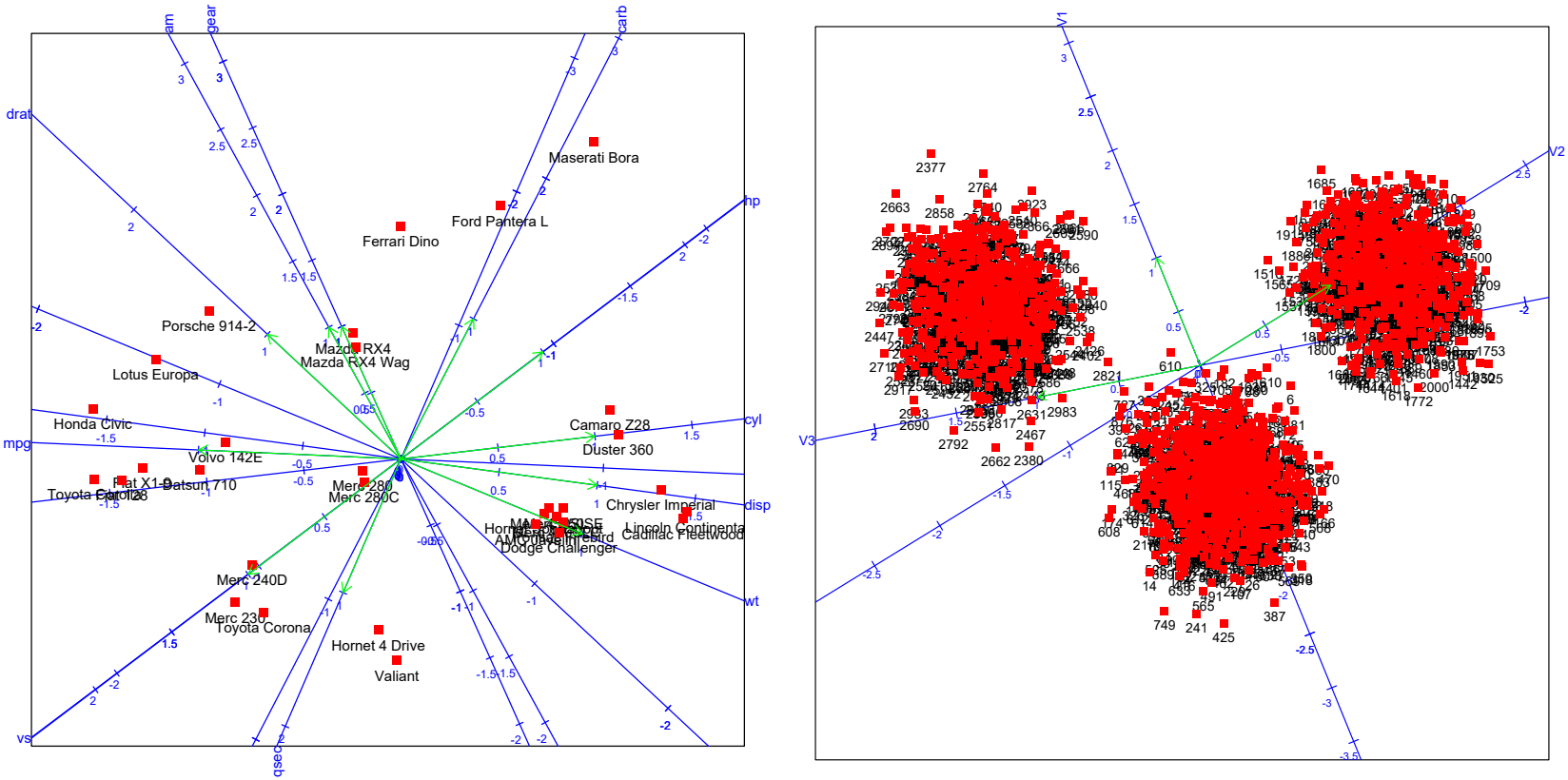


(a) `factoextra`: `mtcars` data set



(b) `factoextra`: `xclara` data set

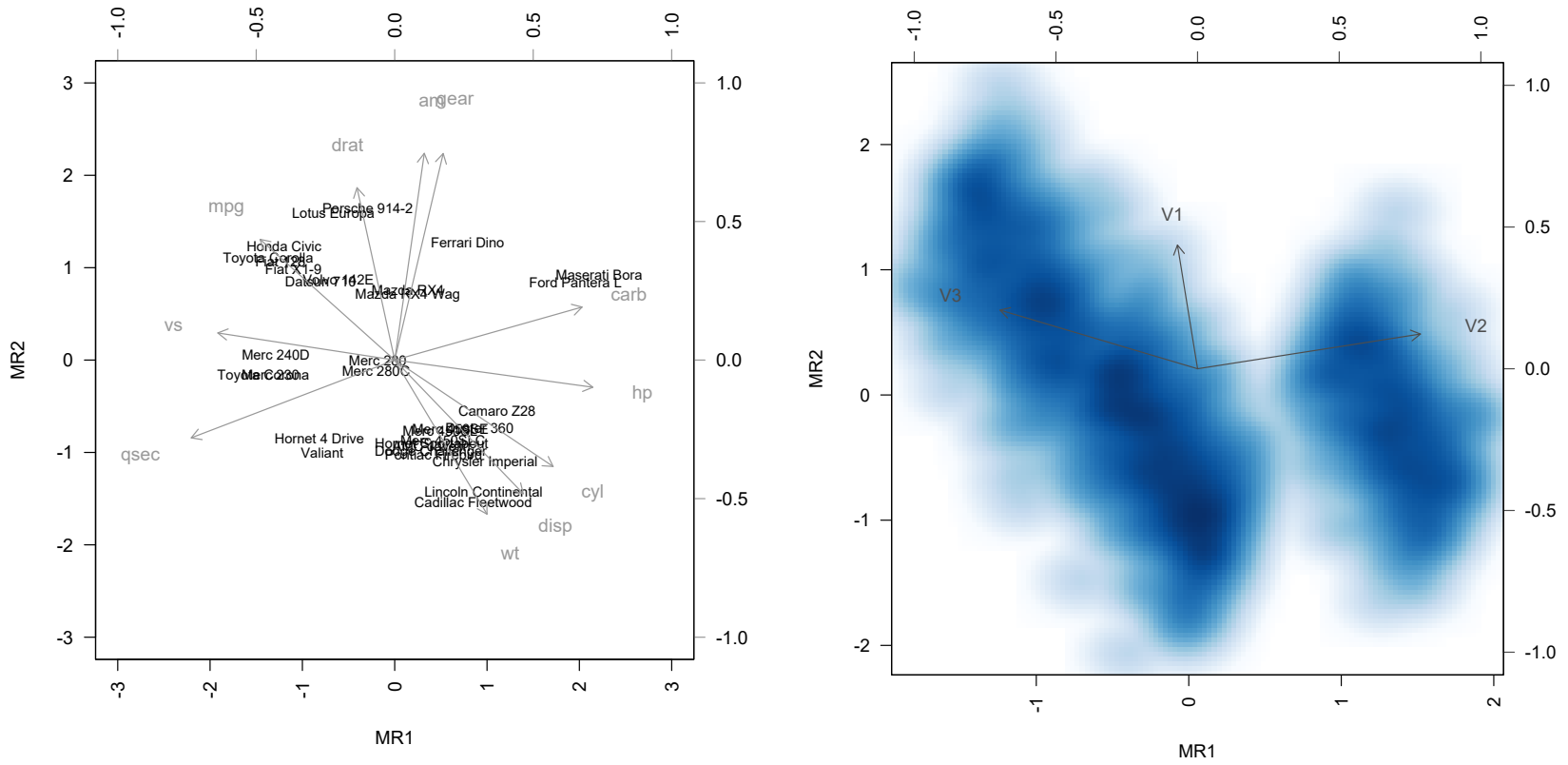
Figure A.3: `factoextra` makes use of `ggplot` instead of base `R` to construct a biplot. It also allows some plotting flexibility not generally found in other packages. For example, there is a parameter `repel` that automates the spacing of object names so that they will not overlap. It further allows the user to either remove variable names or change their features. An example is to convert arrows to points and points into text. The function also allows for concentration ellipses around all the data or in specified groups, as in A.3(b). Although there is plenty of plotting flexibility available to the user, it could still be challenging to see the densities comparatively amongst groups in some instances.



(a) `PLSbiplot1`: `mtcars` data set

(b) `PLSbiplot1`: `xclara` data set

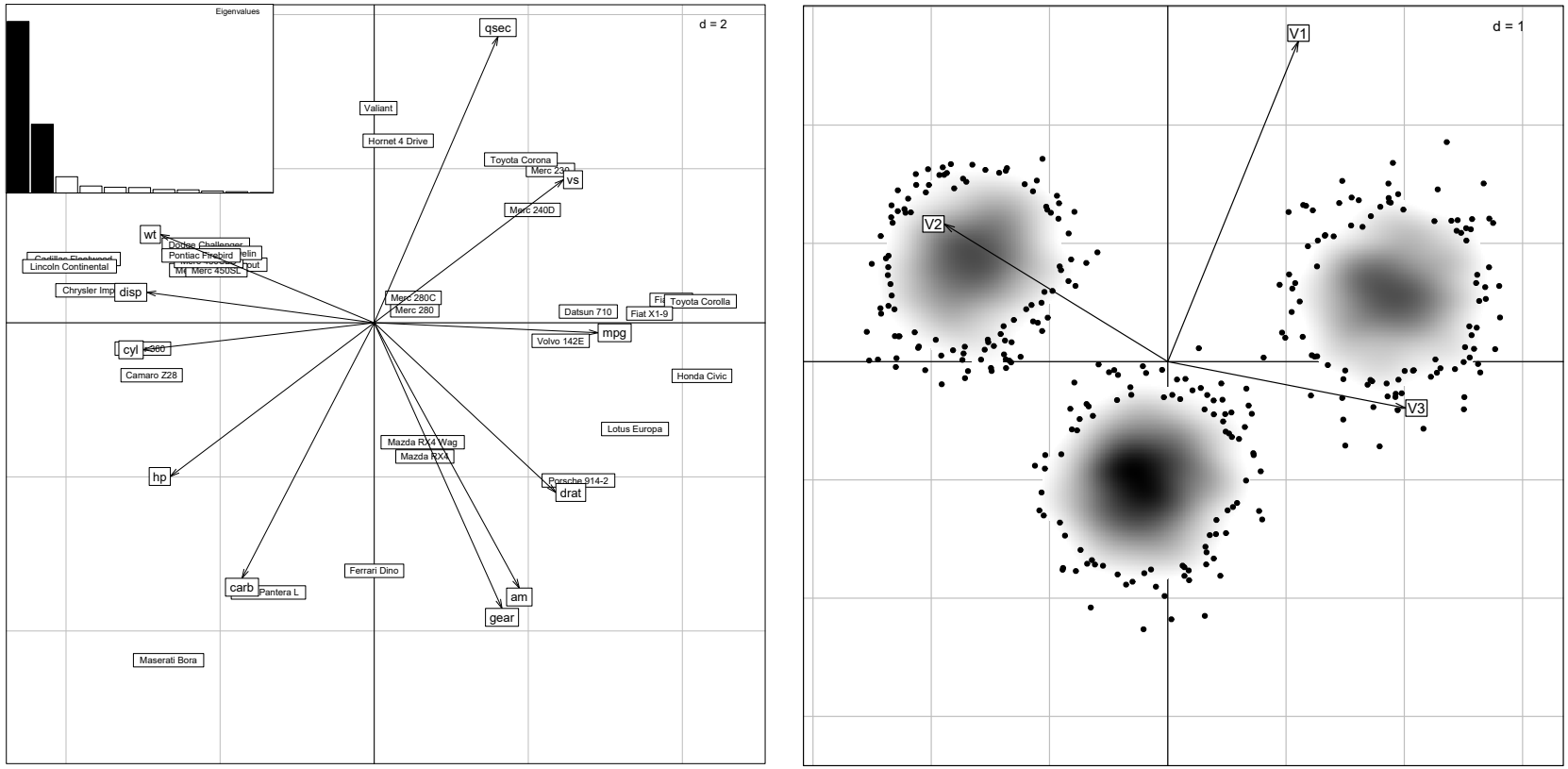
Figure A.4: `PLSbiplot1`, used in the above figures, does not have any parameters to change the visual output. Simultaneous plotting of arrows and axes, generally not available in other biplot packages, are plotted on the biplot. Although these add value to the biplot at first glance, it may make things even more crowded and difficult to interpret.



(a) `psych`: `mtcars` data set

(b) `psych`: `xclara` data set

Figure A.5: The `psych` package, depicted above, has several adjustments to address different issues regarding biplots. It allows the user to replace data points with densities and group variables, allowing the user to differentiate with plotting colours or characters. Simplistic adjustments that are also in many other packages include, adjusting the arrow lengths and colours, removing variable labels, or specifying own labels. What they have done extra is to make provision to exclude specific data labels if they match a criterion, e.g. only show them for points a chosen distance away from the origin. Removing these observation labels can allow for a biplot that lets the user see the essential information whilst not overcrowding the plot.



(a) `ade4`: `mtcars` data set

(b) `ade4`: `xclara` data set

Figure A.6: The `ade4` package, depicted above, allows the user to specify groups or clusters and consequently construct densities of the different groups. These densities are very useful to compare the structure of the data and to see how there relate even though they might look similar when it is not displayed. Furthermore, the `mtcars` data set shows the simple construction of a biplot with a tidy outlay. These plotting functions are coupled with other plotting functions to see, say the vectors separately which could be useful in inspecting specific elements or relationships between elements. This package also has a GUI interface available called `ade4TkGUI` which produces similar plots to the non-interactive package used here.